

# **CS-GY 6923: Lecture 3**

## **Regularization + Bayesian Perspective**

---

NYU Tandon School of Engineering, Akbar Rafiey

- The first written HW has been posted, due Feb 19.
- 10% extra credit will be given if solutions are typewritten using LaTeX, Markdown, etc.
- All solutions must be written independently.
- Reminder: Lab 02 due Feb 12.

## Model selection:

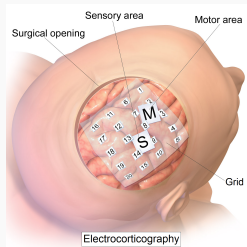
- Train models  $f_{\theta_1}^{(1)}, \dots, f_{\theta_q}^{(q)}$  independently on training data to find optimal parameters  $\theta_1^*, \dots, \theta_q^*$ .
- Check loss  $L_{test}(f^{(1)}, \theta_1^*), \dots, L_{test}(f^{(q)}, \theta_q^*)$  on test data.
- Select model with lowest test loss.

Can be used for arbitrary sets of models. Often used when you are not sure how “complex” your model should be for the data, and want to find a sweet spot between a good fit, and not overfitting.

# Model selection example

## Electrocorticography ECoG (Lab 02):

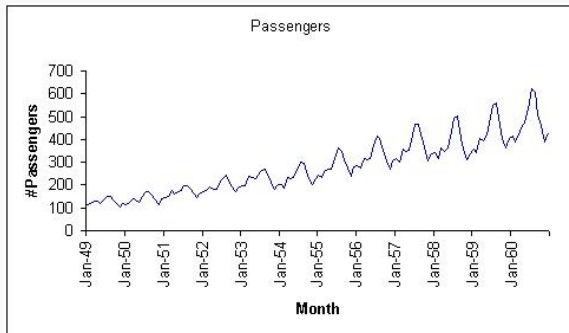
- Implant grid of electrodes on surface of the brain to measure electrical activity in different regions.



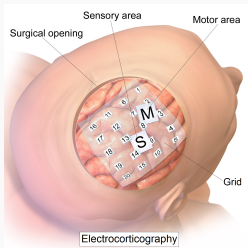
- Predict hand motion based on ECoG measurements.
- **Model order:** predict movement at time  $t$  using brain signals at time  $t, t - 1, \dots, t - q$  for varying values of  $q$ .

## Autoregressive model

Predicting time  $t$  based on a linear function of the signals at time  $t, t - 1, \dots, t - q$  is not the same as fitting a line to the time series. It's much more expressive.



## Electrocorticography ECoG lab:



**First lab where computation actually matters (solving regression problems with  $\sim 40k$  examples,  $\sim 1500$  features)**

Makes sense to test and debug code using a subset of the data.





**Slight caveat: The train-test paradigm is typically not how machine learning or scientific discovery works in practice!**

## Typical workflow:


- Train a class of models.
- Test.
- Adjust class of models.
- Test.
- Adjust class of models.
- Cont...

Final model implicitly depends on test set because performance on the test set guided how we changed our model.

## Popularity of ML benchmarks and competitions leads to adaptivity at a massive scale.

11 Active Competitions		
	<b>Deepfake Detection Challenge</b> Identify videos with facial or voice manipulations <small>Featured · Code Competition · 2 months to go · video data, online video</small>	\$1,000,000 1,595 teams
	<b>Google QUEST Q&amp;A Labeling</b> Improving automated understanding of complex question answer content <small>Featured · Code Competition · 19 hours to go · text data, nlp</small>	\$25,000 1,559 teams
	<b>Real or Not? NLP with Disaster Tweets</b> Predict which Tweets are about real disasters and which ones are not <small>Getting Started · Ongoing · text data, binary classification</small>	\$10,000 2,657 teams
	<b>Bengali.AI Handwritten Grapheme Classification</b> Classify the components of handwritten Bengali <small>Research · Code Competition · a month to go · multiclass classification, image data</small>	\$10,000 1,194 teams

Kaggle (various competitions)

 14,197,122 images, 21841 synsets indexed

[Explore](#) [Download](#) [Challenges](#) [Publications](#) [Updates](#) [About](#)

Not logged in. [Login](#) | [Signup](#)

Imagenet (image classification and categorization)



## Is adaptivity a problem? Does it lead to over-fitting? How much? How can we prevent it?

REPORT

### The reusable holdout: Preserving validity in adaptive data analysis

Cynthia Dwork<sup>1,\*</sup>, Vitaly Feldman<sup>2,\*</sup>, Moritz Hardt<sup>3,\*</sup>, Toniann Pitassi<sup>4,\*</sup>, Omer Reingold<sup>5,\*</sup>, Aaron Roth<sup>6,\*</sup>

+ See all authors and affiliations

Science 07 Aug 2015:  
Vol. 349, Issue 6248, pp. 636-638  
DOI: 10.1126/science.aaa9375

### Do ImageNet Classifiers Generalize to ImageNet?

Benjamin Recht\*  
UC Berkeley

Rebecca Roelofs  
UC Berkeley

Ludwig Schmidt  
UC Berkeley

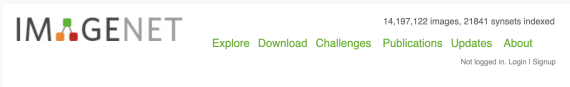
Vaishal Shankar  
UC Berkeley

#### Abstract

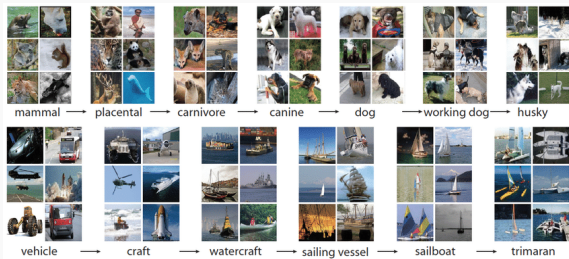
We build new test sets for the CIFAR-10 and ImageNet datasets. Both benchmarks have been the focus of intense research for almost a decade, raising the danger of overfitting to excessively re-used test sets. By closely following the original dataset creation processes, we test to what extent current classification models generalize to new data. We evaluate a broad range of models and find accuracy drops of 3% – 15% on CIFAR-10 and 11% – 14% on ImageNet. However, accuracy gains on the original test sets translate to larger gains on the new test sets. Our results suggest that the accuracy drops are not caused by adaptivity, but by the models' inability to generalize to slightly "harder" images than those found in the original test sets.

12 Jun 2019

# Imagenet dataset

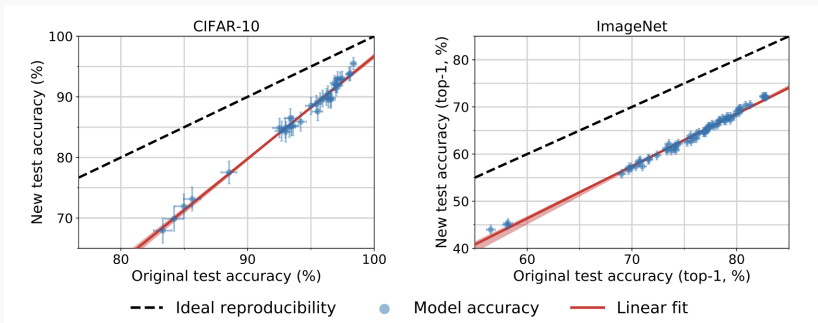


Collected by Fei-Fei Li's group at Stanford in 2006ish and labeled using Amazon Mechanical Turk.



We now have neural network models that can solve these classification problems with  $> 95\%$  accuracy.

## Do ImageNet Classifiers Generalized to ImageNet?



Interestingly, when comparing popular vision models on “fresh” data, while performance dropped across the board, the relative rank of model performance did not change significantly.

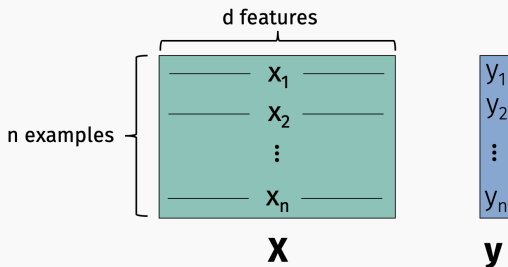
# Regularization

## Over-parameterized models

- In the model selection examples we discussed last class, we had full control over the complexity of the model: could range from underfitting to overfitting.
- In practice, you often don't have this freedom. Even the most basic model might lead to overfitting.

# Over-parameterized models

**Example:** Linear regression model where  $d \geq n$ .

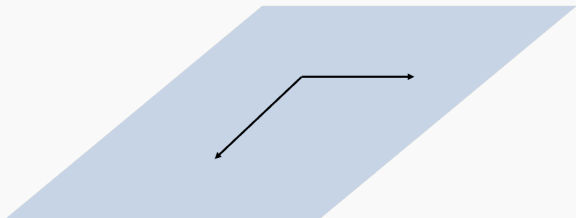


Can (almost) always find  $\beta$  so that  $\mathbf{X}\beta = \mathbf{y}$  exactly.

Why?

## High dimensional linear models

**Claim:** For almost all sets of  $n$ , length  $n$  vectors  $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}$ , we can write any vector  $\mathbf{y}$  as a linear combination of these vectors.



I.e., can find some coefficients so that

$$\beta_1 \mathbf{x}^{(1)} + \dots + \beta_d \mathbf{x}^{(d)} = \mathbf{X}\boldsymbol{\beta} = \mathbf{y}.$$

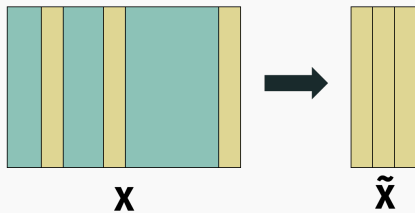
## Zero train loss

- We will discuss some models later in the class where zero training loss is not necessarily a bad sign:  $k$ -nearest neighbors, some neural nets.
- Typically however it will be a sign of overfitting, as in the polynomial regression example.



## Feature selection

Select some subset of  $\ll n$  features to use in model:



**Filter method:** Compute some metric for each feature, and select features with highest score.

- Example: compute loss or  $R^2$  value when each feature in  $\mathbf{X}$  is used in single variate regression.

**Any potential limitations of this approach?**

**Exhaustive approach:** Pick best subset of  $q$  features. Train  $\binom{d}{q}$  models.

# Feature selection

**Faster approach:** Greedily select  $q$  features.

## Stepwise Regression:

- **Forward:** Step 1: pick single feature that gives lowest loss.  
Step  $k$ : pick feature that when combined with previous  $k - 1$  chosen features gives lowest loss.
- **Backward:** Start with all of the features. Greedily eliminate those which have least impact on model performance.

**Feature selection deserves more than two slides, but we won't go into too much more detail!**

**Regularization:** Discourage overfitting by adding a regularization penalty to the loss minimization problem.

$$\min_{\beta} [L(\beta) + \text{Reg}(\beta)].$$

## Alternative approach

**Regularization:** Discourage overfitting by adding a regularization penalty to the loss minimization problem.

$$\min_{\beta} [L(\beta) + \text{Reg}(\beta)].$$

**Example:** Least squares regression:  $L(\beta) = \|\mathbf{X}\beta - \mathbf{y}\|_2^2$ .

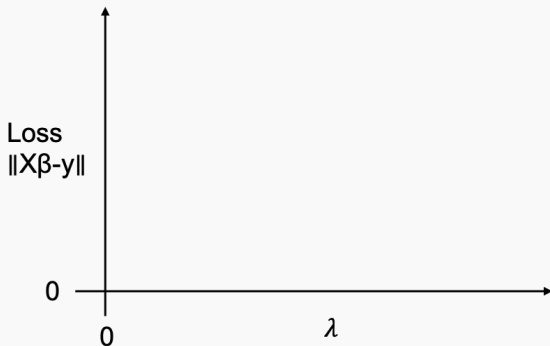
- Ridge regression ( $\ell_2$ ):  $\text{Reg}(\beta) = \lambda \|\beta\|_2^2$
- LASSO (least absolute shrinkage and selection operator) ( $\ell_1$ ):  
 $\text{Reg}(\beta) = \lambda \|\beta\|_1$
- Elastic net:  $\text{Reg}(\beta) = \lambda_1 \|\beta\|_1 + \lambda_2 \|\beta\|_2^2$

Note that  $\arg \min_{\beta} [L(\beta) + \text{Reg}(\beta)] \neq \arg \min_{\beta} [L(\beta)]$

## Ridge regularization: perspective 1

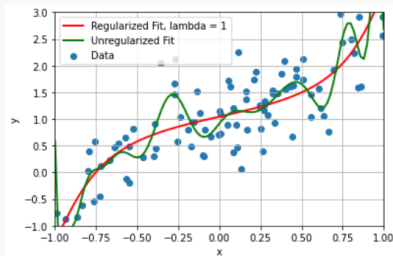
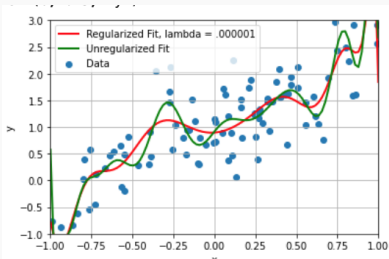
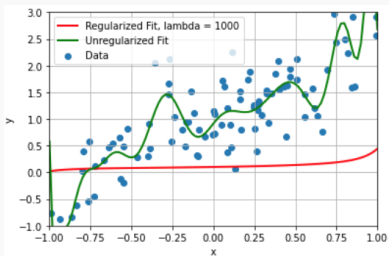
Ridge regression:  $\min_{\beta} (\|\mathbf{X}\beta - \mathbf{y}\|_2^2 + \lambda\|\beta\|_2^2)$ .

- As  $\lambda \rightarrow \infty$ , we expect  $\|\beta\|_2^2 \rightarrow 0$  and  $\|\mathbf{X}\beta - \mathbf{y}\|_2^2 \rightarrow \|\mathbf{y}\|_2^2$ .
- By choosing different values of  $\lambda$  we have models of varying accuracy/complexity.



# Polynomial examples

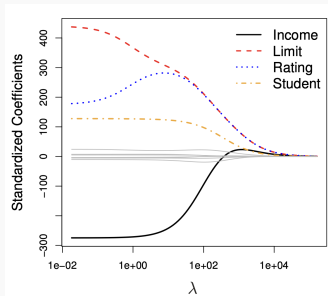
Fit degree 20 polynomial with varying levels of regularization.



## Ridge regularization: perspective 2

Ridge regression:  $\min_{\beta} (\|\mathbf{X}\beta - \mathbf{y}\|_2^2 + \lambda\|\beta\|_2^2)$ .

- As  $\lambda \rightarrow \infty$ , we expect  $\|\beta\|_2^2 \rightarrow 0$  and  $\|\mathbf{X}\beta - \mathbf{y}\|_2^2 \rightarrow \|\mathbf{y}\|_2^2$ .
- Feature selection methods attempt to set many coordinates in  $\beta$  to 0. Ridge regularizations encourages coordinates to be close to zero.





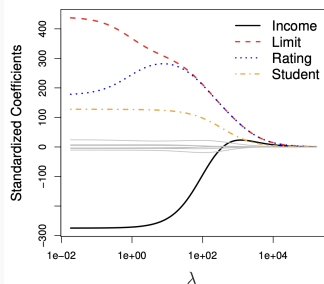
## Ridge regularization

How do we minimize:  $L_R(\beta) = \|\mathbf{X}\beta - \mathbf{y}\|_2^2 + \lambda\|\beta\|_2^2$ ?

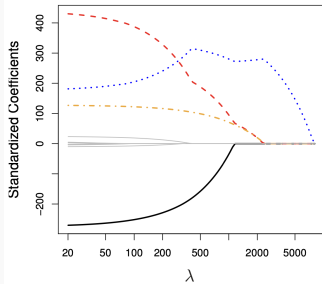
# Lasso regularization

Lasso regularization:  $\min_{\beta} \|\mathbf{X}\beta - \mathbf{y}\|_2^2 + \lambda\|\beta\|_1$ .

- As  $\lambda \rightarrow \infty$ , we expect  $\|\beta\|_1 \rightarrow 0$  and  $\|\mathbf{X}\beta - \mathbf{y}\|_2^2 \rightarrow \|\mathbf{y}\|_2^2$ .
- Typically encourages subset of  $\beta_i$ 's to go to zero, in contrast to ridge regularization.



Ridge regularization



Lasso Regularization

# Lasso regularization

## Pros:

- Simpler, more interpretable model.
- More intuitive reduction in model order.

## Cons:

- No closed form solution because  $\|\beta\|_1$  is not differentiable.
- Can be solved with iterative methods, but generally not as quickly as ridge regression.

## Notes:

- Model selection/cross validation used to choose optimal scaling  $\lambda$  on  $\lambda\|\beta\|_2^2$  or  $\lambda\|\beta\|_1$ .
- Often grid search for best parameters is performed in “log space”. E.g. consider  $[\lambda_1, \dots, \lambda_q] = 1.5^{[-4, -3, -2, -1, -0, 1, 2, 3, 4]}$ .
- Regularization methods are not invariant to data scaling. Typically when using regularization we mean center and scale columns to have unit variance.

**The bayesian/probabilistic modeling perspective**

# Classification setup

- **Data Examples:**  $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$
- **Target:**  $y_1, \dots, y_n \in \{0, 1, \dots, q - 1\}$  when there are  $q$  classes.
  - Binary Classification:  $q = 2$ , so each  $y_i \in \{0, 1\}$ .
  - Multi-class Classification:  $q > 2$ .<sup>1</sup>

---

<sup>1</sup>Note that there is also multi-label classification where each data example may belong to more than one class.

## Classification examples

- Medical diagnosis from MRI: 2 classes.
- MNIST digits: 10 classes.
- Full Optical Character Recognition: 100s of classes.
- ImageNet challenge: 21,000 classes.

**Running example today: Email Spam Classification.**

# Classification

Classification can (and often is) solved using the same **loss-minimization framework** we saw for regression.

We won't see that today! We're going to use classification as a window into another way of thinking about machine learning.

Will give an interesting new justifications for tools like regularization. Will also give us an approach for generative ML.

**Rest of Today:** ML from a **Probabilistic Modeling/Bayesian Perspective**.



# Probabilistic modeling

In a Bayesian or Probabilistic approach to machine learning we always start by conjecturing a

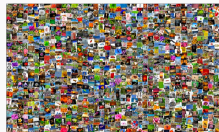
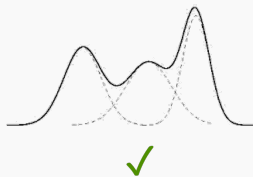
## probabilistic model

that plausibly could have generated our data.

- The model guides how we make predictions.
- The model typically has unknown parameters  $\vec{\theta}$  and we try to find the most reasonable parameters based on **observed** data (more on this later in lecture).

# Probabilistic modeling

Typically we try to keep things simple!



## Probabilistic modeling

**Exercise:** Come up with a probabilistic model for the following data set  $(x_1, y_1), \dots, (x_n, y_n)$ .

- For  $n$  **NYC apartments**: each  $x_i$  is the size of the apartment in square feet. Each  $y_i$  is the monthly rent in dollars.

What are the unknown parameters of your model. What would be a guess for their values? How would you confirm or refine this guess using data?

# Probabilistic modeling

**Dataset:**  $(x_1, y_1), \dots, (x_n, y_n)$

**Description:** For  $n$  **NYC apartments**: each  $x_i$  is the size of the apartment in square feet. Each  $y_i$  is the monthly rent in dollars.

**Model:**

# Probabilistic modeling

**Dataset:**  $(x_1, y_1), \dots, (x_n, y_n)$

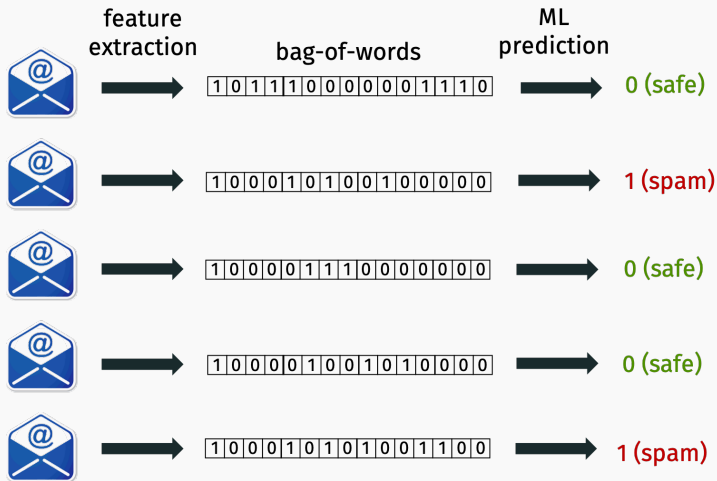
**Description:** For  $n$  **students**: each  $x_i \in \{\textit{Fresh.}, \textit{Soph.}, \textit{Jun.}, \textit{Sen.}\}$  indicating class year. Each  $y_i \in \{0, 1\}$  with zero indicating the student has not taken machine learning, one indicating they have.

**Model:**

## Goal:

- Build a probabilistic model for a binary classification problem.
- Estimate parameters of the model.
- From the model derive a classification rule for future predictions (the **Naive Bayes Classifier**).

# Spam prediction



Both target labels and data vectors are binary.

# Email model

Let's create a model that generates spam and non-spam emails.

**Observation:** Since bag-of-words features don't care about word order, our model does not need to either.

- Common approach: assign a probability  $p_i \in [0, 1]$  to word  $i$ .  
Set  $\mathbf{x}_i = 1$  with probability  $p_i$ ,  $\mathbf{x}_i = 0$  with probability  $1 - p_i$ .

$p_{\text{the}} =$

$p_{\text{calendar}} =$

$p_{\text{toothbrush}} =$



# Email model

The screenshot shows an email client interface with a search bar at the top containing the text "toothbrush". Below the search bar are filter buttons: "Mail", "Conversations", "Spaces", "From", "Any time", "Has attachment", "To", "Exclude Promotions", "Is unread", and "Advanced". The main area displays a list of 17 email results, each with a sender icon, sender name, subject, and date. Several subject lines contain the word "toothbrush" highlighted in yellow.

Sender	Subject	Date
Amazon Marketplace	akbar, will you rate your transaction at Amazon.com? - image" align="top" ...	
Amazon.com	Your Amazon.com order of "mopio Futon Sofa Bed..." and 10 more items. - Amazon.com Order Confir...	Sep 10
Amazon.com	<b>The Labor Day Sale is here</b> - nas Colgate <b>Toothbrush</b> Limited time deal -35% \$7.75 List Price: \$11.99...	<b>Aug 26</b>
Instacart	Your Instacart order receipt - Thanks for ordering from Instacart! Your order from CVS® was delivered...	May 5
Amazon.com	<b>Up to 40% off last-minute gifts</b> - cl_5_3_manu_img_b Electric <b>toothbrushes</b> Give them (or you) the ...	<b>12/17/23</b>
Instacart	<b>You left an item behind...</b> - You left an item behind... 96 @import url("https://links.customers.i...	<b>12/3/23</b>
Amazon.ca	<b>Shop epic Black Friday deals</b> - Rechargeable Electric <b>Toothbrush</b> , White, HX6817/01 by Philips Sonic...	<b>11/24/23</b>
Amazon.com Reviews	akbar, did 'ALIVER Nail Polish Remover' meet your expectations? <b>Review it on Amazon</b> - 2CB07...	<b>10/14/23</b>
Amazon.com Reviews	akbar, did 'FACEMADE Nail Clippers Set' meet your expectations? <b>Review it on Amazon</b> - 2CB0...	<b>10/14/23</b>
Amazon Marketplace	akbar, will you rate your transaction at Amazon.com? - image" align="top" height="60" border="0" /> ...	9/9/23
Amazon.com	Your Amazon.com order of "WLIVE Coffee Table Set of..." and 1 more item. - Amazon.com Order Confir...	9/5/23
Amazon.com	<b>Hooray, it's Prime Day!</b> - Philips Sonicare <b>Toothbrush</b> by Philips Sonicare https://www.amazon.com/g...	<b>7/11/23</b>
UNHCR Canada	Last chance to give more than a gift - gift of <b>toothbrushes</b> , toothpaste, towels, soap and jerry cans to...	12/22/22
Uber Eats	<b>Running low on essentials?</b> - Get \$12 off a \$25+ order of hand soap, <b>toothbrushes</b> , and more.	<b>7/7/22</b>
Amazon.ca	<b>Deals have landed for Cyber Monday!</b> - Sonicare Electric <b>Toothbrushes</b> & Air... https://www.amazon...	<b>11/27/21</b>
Simon Fraser Dental	Referral Prize - complimentary electric <b>toothbrush</b> . Refer A Friend & Receive A FREE Oral-B Vitality Re...	4/28/18

# Email model

How can we make this model richer when we take spam into account?

- Different words tend to be more or less frequent in spam or regular emails.

Not Spam

$$p_{\text{won}} =$$

$$p_{\$} =$$

$$p_{\text{student}} =$$

Spam

$$p_{\text{won}} =$$

$$p_{\$} =$$

$$p_{\text{student}} =$$

# Probabilistic model for email

**Probabilistic model** for (bag-of-words, label) pair  $(\mathbf{x}, y)$ :

- Set  $y = 0$  with probability  $p_0$ ,  $y = 1$  with probability  $p_1 = 1 - p_0$ .
  - $p_0$  is probability an email is not spam (e.g. 99%).
  - $p_1$  is probability an email is spam (e.g. 1%).
- If  $y = 0$ , for each  $i$ , set  $x_i = 1$  with prob.  $p_{i0}$ .
- If  $y = 1$ , for each  $i$ , set  $x_i = 1$  with prob.  $p_{i1}$ .

**Unknown model parameters:**

- $p_0, p_1$ ,
- $p_{10}, p_{20}, \dots, p_{d0}$ , one for each of the  $d$  vocabulary words.
- $p_{11}, p_{21}, \dots, p_{d1}$ , one for each of the  $d$  vocabulary words.

**How would you estimate these parameters?**

## Reasonable way to set parameters:

- Set  $p_0$  and  $p_1$  to the empirical fraction of not spam/spam emails.
- For each word  $i$ , set  $p_{i0}$  to the empirical probability word  $i$  appears in a non-spam email.
- For each word  $i$ , set  $p_{i1}$  to the empirical probability word  $i$  appears in a spam email.

**Estimating these parameters from previous data examples is the only “training” we will do.**

Done with modeling  
on to prediction

## Probability review

- **Probability:**  $p(x)$  – the probability event  $x$  happens.
- **Joint probability:**  $p(x,y)$  – the probability that event  $x$  and event  $y$  happen.
- **Conditional Probability**  $p(x | y)$  – the probability  $x$  happens given that  $y$  happens.

$$p(x|y) =$$

## Bayes theorem/rule

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}$$

**Proof:**

## Classification rule

Given unlabeled input  $(\mathbf{w}, \text{---})$ , choose the label  $y \in \{0, 1\}$  which is most likely given the data. Recall  $\mathbf{w} = [0, 0, 1, \dots, 1, 0]$ .

**Classification rule:** **maximum a posterior (MAP) estimate.**

**Step 1. Compute:**

- $p(y = 0 \mid \mathbf{w})$ : prob.  $y = 0$  given observed data vector  $\mathbf{w}$ .
- $p(y = 1 \mid \mathbf{w})$ : prob.  $y = 1$  given observed data vector  $\mathbf{w}$ .

**Step 2. Output:** 0 or 1 depending on which probability is larger.

$p(y = 0 \mid \mathbf{w})$  and  $p(y = 1 \mid \mathbf{w})$  are called **posterior** probabilities.



## Evaluating the posterior

How to compute the posterior? **Bayes rule!**

$$p(y = 0 | \mathbf{w}) = \frac{p(\mathbf{w} | y = 0)p(y = 0)}{p(\mathbf{w})} \quad (1)$$

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}} \quad (2)$$

- **Prior:** Probability in class 0 prior to seeing any data.
- **Posterior:** Probability in class 0 after seeing the data.

## Evaluating the posterior

Goal is to determine which is larger:

$$p(y = 0 | \mathbf{w}) = \frac{p(\mathbf{w} | y = 0)p(y = 0)}{p(\mathbf{w})} \quad \text{vs.}$$
$$p(y = 1 | \mathbf{w}) = \frac{p(\mathbf{w} | y = 1)p(y = 1)}{p(\mathbf{w})}$$

- We can ignore the evidence  $p(\mathbf{w})$  since it is the same for both sides!
- $p(y = 0)$  and  $p(y = 1)$  already known (computed from training data). These are our computed parameters  $p_0, p_1$ .
- $p(\mathbf{w} | y = 0) = ?$   $p(\mathbf{w} | y = 1) = ?$

## Evaluating the posterior

Consider the example  $\mathbf{w} = [0, 1, 1, 0, 0, 0, 1, 0]$ .

Recall that, under our model, index  $i$  is 1 with probability  $p_{i0}$  if we are not spam, and 1 with probability  $p_{i1}$  if we are spam .

$$p(\mathbf{w} \mid y = 0) =$$

$$p(\mathbf{w} \mid y = 1) =$$

## Final Naive Bayes Classifier

**Training/Modeling:** Use existing data to compute:

- $p_0 = p(y = 0), p_1 = p(y = 1)$
- For all  $i$  compute:
  - $p_{i0} = p(x_i = 1 | y = 0)$  and  $(1 - p_{i0}) = p(x_i = 0 | y = 0)$
  - $p_{i1} = p(x_i = 1 | y = 1)$  and  $(1 - p_{i1}) = p(x_i = 0 | y = 1)$

**Prediction:**

- For new input  $\mathbf{w}$ :
  - Compute  $p(\mathbf{w} | y = 0) = \prod_i p(w_i | y = 0)$
  - Compute  $p(\mathbf{w} | y = 1) = \prod_i p(w_i | y = 1)$
- Return

$$\arg \max [p(\mathbf{w} | y = 0) \cdot p(y = 0), p(\mathbf{w} | y = 1) \cdot p(y = 1)].$$

**Other applications of  
the bayesian perspective**

# Bayesian regression

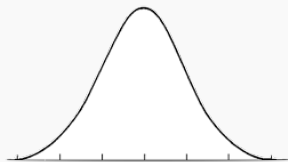
The Bayesian view offers an interesting alternative perspective on many machine learning techniques.

**Example:** Linear Regression.

**Probabilistic model:**

$$y = \langle \mathbf{x}, \boldsymbol{\beta} \rangle + \eta$$

where the  $\eta$  drawn from  $N(0, \sigma^2)$  is **random Gaussian noise**.



$$\Pr(\eta = z) \sim$$

The symbol  $\sim$  means “is proportional to”.

## Gaussian distribution refresher

**Names for the same thing:** Normal distribution, Gaussian distribution, bell curve.

Parameterized by **mean**  $\mu$  and **variance**  $\sigma^2$ .

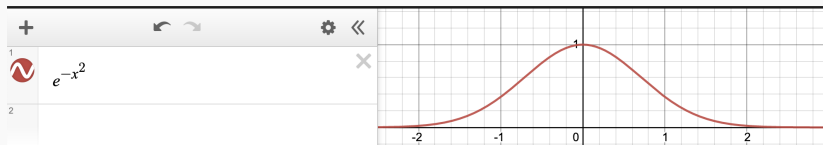


$\eta$  is a continuous random variable, so it has a probability density function  $p(\eta)$  with  $\int_{-\infty}^{\infty} p(\eta) d\eta = 1$

$$p(\eta) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\eta-\mu}{\sigma}\right)^2}$$

## Gaussian distribution refresher

The important thing to remember is that the the PDF falls off exponentially as we move further from the mean.



The normalizing constant in front  $1/2$ , etc. don't matter so much.



**Example:** Linear Regression.

**Probabilistic model:**

$$y = \langle \mathbf{x}, \boldsymbol{\beta} \rangle + \eta$$

where the  $\eta$  drawn from  $N(0, \sigma^2)$  is **random Gaussian noise**. The noise is independent for different inputs  $\mathbf{x}_1, \dots, \mathbf{x}_n$ .

## How should we select $\beta$ for our model?

Also use a Bayesian approach!

Choose  $\beta$  to maximize:

$$\text{posterior} = \Pr(\beta \mid \mathbf{X}, \mathbf{y}) = \frac{\Pr(\mathbf{X}, \mathbf{y} \mid \beta) \Pr(\beta)}{\Pr(\mathbf{X}, \mathbf{y})} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}.$$

In this case, we don't have a prior – no values of  $\beta$  are inherently more likely than others.

Choose  $\beta$  to maximize just the likelihood:

$$\frac{\Pr(\mathbf{X}, \mathbf{y} \mid \beta) \Pr(\beta)}{\Pr(\mathbf{X}, \mathbf{y})} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}.$$

This is called the **maximum likelihood estimate**.

## Fixed design linear regression

Often we think of  $\mathbf{X}$  as fixed and deterministic, and only  $\mathbf{y}$  is generated at random in the model. This is called the fixed design setting. Can also consider a randomized design setting, but it is slightly more complicated.

In the fixed design setting our task of maximizing  $\Pr(\mathbf{X}, \mathbf{y} \mid \beta)$  simplifies to maximizing

$$\max_{\beta} \Pr(\mathbf{y} \mid \beta)$$

# Maximum likelihood estimate

**Data:**

$$\mathbf{X} = \begin{bmatrix} - & \mathbf{x}_1 & - \\ - & \mathbf{x}_2 & - \\ & \vdots & \\ - & \mathbf{x}_n & - \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

**Model:**  $y_i = \langle \mathbf{x}_i, \boldsymbol{\beta} \rangle + \eta_i$  where  $p(\eta_i = z) \sim e^{-z^2/2\sigma^2}$  and  $\eta_1, \dots, \eta_n$  are independent.

$$\Pr(\mathbf{y} \mid \boldsymbol{\beta}) \sim$$

Easier to work with the **log likelihood**:

$$\begin{aligned}\arg \max_{\beta} \Pr(\mathbf{X}, \mathbf{y} \mid \beta) &= \arg \max_{\beta} \prod_{i=1}^n e^{-(y_i - \langle \mathbf{x}_i, \beta \rangle)^2 / 2\sigma^2} \\ &= \arg \max_{\beta} \log \left( \prod_{i=1}^n e^{-(y_i - \langle \mathbf{x}_i, \beta \rangle)^2 / 2\sigma^2} \right) \\ &= \arg \max_{\beta} \sum_{i=1}^n -(y_i - \langle \mathbf{x}_i, \beta \rangle)^2 / 2\sigma^2 \\ &= \arg \min_{\beta} \sum_{i=1}^n (y_i - \langle \mathbf{x}_i, \beta \rangle)^2.\end{aligned}$$

**Conclusion:** Choose  $\beta$  to minimize:

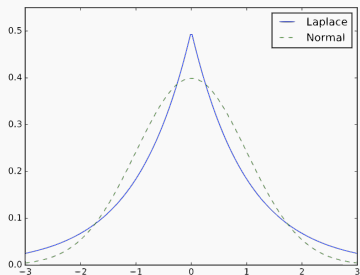
$$\sum_{i=1}^n (y_i - \langle \mathbf{x}_i, \beta \rangle)^2 = \|\mathbf{y} - \mathbf{X}\beta\|_2^2.$$

**This is a completely different justification for squared loss!**

Minimizing the  $\ell_2$  loss is optimal in a certain sense when you assume your data follows a linear model with i.i.d. Gaussian noise.

# Bayesian regression

If we had modeled our noise  $\eta$  as Laplace noise, we would have found that minimizing  $\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_1$  was optimal.



$$Pr(\eta = z) \sim$$

Laplace noise has “heavier tails”, meaning that it results in more outliers.

**This is a completely different justification for  $\ell_1$  loss.**

## Bayesian regularization

We can add another layer of probabilistic modeling by also assuming  $\beta$  is random and comes from some distribution, which encodes our prior belief on what the parameters are.

Maximum a posteriori (MAP estimation):

$$\Pr(\beta \mid \mathbf{X}, \mathbf{y}) = \frac{\Pr(\mathbf{X}, \mathbf{y} \mid \beta) \Pr(\beta)}{\Pr(\mathbf{X}, \mathbf{y})}.$$

Assume values in  $\beta = [\beta_1, \dots, \beta_d]$  come from some distribution.

- **Common model:** Each  $\beta_i$  drawn from  $N(0, \gamma^2)$ , i.e. normally distributed, independent.
- Encodes a belief that we are unlikely to see models with very large coefficients.



# Bayesian regularization

**Goal:** choose  $\beta$  to maximize:

$$\Pr(\beta \mid \mathbf{X}, \mathbf{y}) = \frac{\Pr(\mathbf{X}, \mathbf{y} \mid \beta) \Pr(\beta)}{\Pr(\mathbf{X}, \mathbf{y})}.$$

- We can still ignore the “evidence” term  $\Pr(\mathbf{X}, \mathbf{y})$  since it is a constant that does not depend on  $\beta$ .
- $\Pr(\beta) = \Pr(\beta_1) \cdot \Pr(\beta_2) \cdot \dots \cdot \Pr(\beta_d)$
- $\Pr(\beta) \sim$

# Bayesian regularization

Easier to work with the **log likelihood**:

$$\begin{aligned} & \arg \max_{\beta} \Pr(\mathbf{X}, \mathbf{y} \mid \beta) \cdot \Pr(\beta) \\ &= \arg \max_{\beta} \prod_{i=1}^n e^{-(y_i - \langle \mathbf{x}_i, \beta \rangle)^2 / 2\sigma^2} \cdot \prod_{i=1}^n e^{-(\beta_i)^2 / 2\gamma^2} \\ &= \arg \max_{\beta} \sum_{i=1}^n -(y_i - \langle \mathbf{x}_i, \beta \rangle)^2 / 2\sigma^2 + \sum_{i=1}^d -(\beta_i)^2 / 2\gamma^2 \\ &= \arg \min_{\beta} \sum_{i=1}^n (y_i - \langle \mathbf{x}_i, \beta \rangle)^2 + \frac{\sigma^2}{\gamma^2} \sum_{i=1}^d (\beta_i)^2 / \sigma^2. \end{aligned}$$

Choose  $\beta$  to minimize  $\|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \frac{\sigma^2}{\gamma^2} \|\beta\|_2^2$ .

**Completely different justification for ridge regularization!**

**Test your intuition:** What modeling assumption justifies LASSO regularization:  $\min \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1$ ?